The ITOS Data Point Server

Overview

The ITOS data point server allows network clients to obtain live telemetry mnemonic values from an ITOS system. Clients for the server include the Java-based telemetry page displays.

The server is a single-thread event-driven application. When the server starts, it creates a TCP/IP socket and waits for clients to connect to the socket over a well-known-port. The socket is protected by TCP Wrappers and requires an entry in the file '`/etc/hosts.allow`' defined as '*dp_server*' with the list of fully or partially qualified hosts names allowed access to this server. The local host is allowed access by default and doesn't need to be in this list. See the '`man tcpd`' pages on how to set up access control.

Once a connection has been established, a client makes requests using ascii commands:

ADMIN	Remote control the server.
EXIT	Terminate this connection.
LET	Set a mnemonic's value.
PING	Verify that the server is responding.
SHO	Request a mnemonic value.
TAG	Tag mnemonics so the server can send asynchronous values.
UNTAG	Tells the server to stop sending asynchronous values.

The server responds to the client with ascii messages:

OK	Acknowleges that a request was received.
?	Error message.
v	Mnemonic value.

There are three levels of client privilege:

Clients running on the same host as the server may issue the ADMIN requests - no other clients may issue these requests.

Clients running on the same cluster as the server may use threshold values as small as 0.1 seconds. They may also TAG mnemonics with the 'a' or 'c' flag and LET mnemonic values into the database. To define a cluster you must create an entry in '`/etc/hosts.allow`' named '*dps_cluster*' with list of the fully or partially qualified host names. The local host is checked seperately and doesn't need to be put in this list.

Clients not in the server's cluster may use a threshold of no less than 5 seconds otherwise 5 seconds gets used instead.

The server is unforgiving and breaks the connection to clients that send garbled or unauthorized requests.

1 Client Requests

1.1 ADMIN

When issued from a client on *localhost* (the host where the server is running):

`ADMIN QC`	Show the list of clients.
`ADMIN QC` *client*	Show which mnemonics the *client* has tagged.
`ADMIN QM`	Show which mnemonics are tagged.
`ADMIN QM` *mnemonic*	Show which clients have tagged *mnemonic*.
`ADMIN QV`	Show the current verbosity level.
`ADMIN V` *level*	Change verbosity to *level*.
`ADMIN K` *client*	Kill *client*.

1.2 EXIT

The EXIT request causes the server to break the connection with the client.

It is not necessary to use the EXIT request; it is expected that most clients will simply break the connection.

1.3 LET

The LET request allows privileged clients to assign values to mnemonics. Non-privileged clients (clients not listed in '`/etc/hosts.allow`' *dps_cluster* entry) may not issue LET requests.

The syntax of the LET request is:

```
<request>: LET <mnemonic> <raw-value>
```

mnemonic is the name of the mnemonic, *raw-value* is an appropriate value for the mnemonic. If *raw-value* is a string, it may be in double quotes. Double quotes are required for strings that contain white-space characters (space or tab).

For example:

```
let gbl_ui 12345
⇒ ok
let gbl_char "this is a string"
⇒ ok
let gbl_ui_array[20] 12345
⇒ ok
```

1.4 PING

A PING request consists of the word '`PING`' in upper, lower, or mixed case. The PING request allows clients to verify that the server is alive and well.

The server responds to a ping request with an OK response.

```
ping
⇒ ok
```

1.5 SHO

The SHO request obtains a mnemonic value synchronously. (See the TAG request for how to request mnemonic values asynchronously).

The server responds to a SHO request with either a v response or a ? response.

The syntax of the SHO request is:

```
<request>: SHO <mnemonic>[/[<raw-format>]/<convert-format>][, ...]
```

For example:

```
sho gbl_gmtoff
⇒ v GBL_GMTOFF iG i 0 97-101-17:44:23.016790
sho gbl_gmtoff/"%d"/"%24[%C]D"
⇒ v|GBL_GMTOFF|iG|i|Fri Apr 11 17:44:23 1997
sho h001time/"%24[%C]D"
⇒ v|H001_TIME|d|i|Fri Apr 11 17:44:23 1997
sho psbatcurr/"%u"/"%e"
⇒ v PSBATCURR uA R 22 1.40122e+01
sho gbl_ui_array[1]
⇒ v GBL_UI_ARRAY[1] u i 0x0
```

mnemonic is the mnemonic name.

raw-format and *convert-format* are optional format specifications. **Caution:** These are no longer fully supported so beware.

Missing format strings are inferred from the mnemonic's value type; see the second field of the `v` response for the default format strings.

1.6 TAG

The TAG request allows clients to request mnemonic values asynchronously. (See the SHO request for how to request mnemonic values synchronously).

By asynchronous, we mean that the server sends a value each time some triggering event (such as 'the value changed') occurs. The server keeps sending values until the client breaks the connection or issues an UNTAG request.

The syntax of the TAG request is:

```
<request>: TAG <flag> <threshold> <mnemonic>[/<rfmt>]/<cfmt>][, ...]
```

flag is one of:

`a`	The data point server should send the value each time a new value arrives in the telemetry stream, even if the new value is the same as the old value.
`c`	The data point server should send the value each time the value changes.
`i`	The data point server should send the value on a fixed interval (every *threshold seconds*), without regard to whether or not a new value arrived in the telemetry stream.

Note: clients not in the same cluster as the server don't have permission to use the 'a' or 'c' flags; for these clients, all flags are treated as 'i'.

threshold is the minimum interval (in seconds) between values. For example, if *flag* is '`a`' and *threshold* is '3' the data point server won't send more than one value every 3 seconds,

even if values arrive in the telemetry stream more frequently. (The data point server will simply discard the unsent values). When *flag* is `i`, *threshold* represents the fixed interval. Clients running on the localhost may use threshold values as small as 0.01 and for flags of 'a' or 'c' a threshold value of zero (0) which means no interval testing is done and no values are discarded. Clients running in the cluster may use threshold values as small as 0.1 and the rest may use a value no lower than 5. If a threshold value lower than allowed for that client is requested, the datapoint server will use it's maximum allowed value instead.

mnemonic[/[*rfmt*]/*cfmt*] is a mnemonic name and format string specification as used in the SHO request. There may be multiple specifications, seperated by commas.

Note that it only makes sense for a client to tag a mnemonic once! If a client tags the same mnemonic multiple times, the final tag supercedes all previous tags.

1.7 UNTAG

The UNTAG request cancels a TAG request. The syntax of the UNTAG request is:

```
<request>: UNTAG <mnemonic>[, ...]
```

2 Server Responses

2.1 OK

```
ping
⇒ ok
```

2.2 ? (error)

```
sho aabbccdd
⇒ ? aabbccdd is not a mnemonic.
sho psbatcurr/"%k"
⇒ ? %k is not a valid format specification.
sho 1 2 3 4 5 6 7
⇒ ? invalid syntax: sho 1 2 3 4 5 6 7
sho gbl_ui_array[1000]
⇒ ? GBL_UI_ARRAY[1000] exceeds max mnemonic index of 99
let gbl_ui "abcdef"
⇒ ? Don't know how to convert string "abcedf" to unsigned.
let gbl_ui 12345
⇒ ? LET only valid for hosts in cluster list!
```

Caution: The server is unforgiving and usually breaks the connection after sending a ? response.

2.3 v (value)

A `v` response might look like:

```
sho gbl_gmtoff
⇒ v gbl_gmtoff iG i 0 97-101-17:44:23.016790
sho psbatcurr/"%u"/"%e"
⇒ v psbatcurr uA R 22 1.40122e+01
sho toauxosc
⇒ v|toauxosc|uD|i|0x0000|ON  |white|black
```

The first character, `'v'`, identifies this as a value message.

The second character is the field seperator, which can be any non-alphanumeric character that doesn't appear elsewhere in the response.

The first field is the mnemonic name; this identifies which mnemonic this value is for.

The second field consists of one or two characters that indicate the value type. The first (or only) of these characters indicates the raw type:

`d` (`C_DATE`). The raw value is a date sec and usec past/before the EPOCH. Negative numbers are before the EPOCH as positive after. The raw is never actually seen and is normalized in ITOS. The default format is `"%22[%y-%j-%T.%f]D"`; values in this format look like "`97-301-02:55:12.000000`".

`f` (`C_FLOAT`). The raw value is a floating point number. The default format is `"%g"`.

`i` (`C_INTEGER`). The raw value is a signed integer. The default format is `"%d"`.

`u` (`C_UNSIGNED`). The raw value is an unsigned integer. The default raw format is `"0x%X"`; values in the default format look like "`0x14A7`".

`s` (`C_STRING`). The raw value is a string. The default format is `"%s"`.

`t` (`C_TIME`). The raw value is a relative time sec and usec. The default format is `"%19[%03d-%02h:%02m:%02s.%06f]D"`; values in this format look like "`301-02:55:12.000000`".

`?` Unknown; something is misconfigured.

When there are two characters, the second character indicates the converted type:

`A` (`CNV_ANALOG`). Analog conversion. The raw value gets converted to a floating point number via a polynomial conversion. The default format is `"%g"`.

`D` (`CNV_DISCRETE`). Discrete conversion. The raw value gets converted to a string, a foreground color, and a background color. The default format is `"%s"`.

`F` (`CNV_FLOAT`). Float conversion. Possibly never used.

`G` (`CNV_DATE`). Date conversion. The raw value gets converted to a date by treating the raw value as the number of seconds since the Unix epoch (midnight at the beginning of the day, Jan 1 1970). The default format is `"%22[%y-%j-%T.%f]D"`; values in this format look like "`97-301-02:55:12.000000`".

`I` (`CNV_INDIRECT`). Indirect conversion. Possibly never used.

`M` (`CNV_MNEM_ID`). Mnemonic ID conversion.

`R` (`CNV_PSEUDO_REAL`). Pseudo real conversion.

`T` (`CNV_TIME`). Time conversion. The raw value gets converted to a time by treating the raw value as the number of seconds elapsed. The default format is `"%10[%S.%06f]T"`; values in this format look like "`200.000000`". Possibly never used.

`?` Unknown; something is misconfigured.

The third field contains one or more flag characters:

`u` The value is unset; there is no value; the *EXISTS flag* is clear. Since it is meaningless to say that an unset value is static or to ask whether an unset value is in limits, the `u` flag always appears by itself.

`s` The value is static, which means that the value should get updated via the telemetry stream but hasn't been updated in such a long time that there's likely to be a problem with the telemetry stream.

`Q` The value has bad quality, which typically means that a transfer frame containing part of the packet that contains the value had a CRC error.

`R` The value is red high. The value is not static unless the `s` flag is also present.

`Y` The value is yellow high. The value is not static unless the `s` flag is also present.

`i` The value is in limits. The value is not static unless the `s` flag is also present.

`y` The value is yellow low. The value is not static unless the `s` flag is also present.

`r` The value is red low. The value is not static unless the `s` flag is also present.

`l` The value is rail low; the value is less than the minimum legal value. Since rail low violations aren't currently detected, this flag can be regarded as a future enhancement.

`L` The value is rail high; the value is greater than the maximum legal value. Since rail high violations aren't currently detected, this flag can be regarded as a future enhancement.

`D` The value triggered a delta violation; the difference between the current value and the previous value is greater than delta. Since delta violations aren't currently detected, this flag can be regarded as a future enhancement.

The fourth field is the raw value.

If there's no converted value (i.e., the second field doesn't have the second character), the message ends after the fourth field.

Otherwise, the fifth field is the converted value. If the mnemonic has a discrete conversion but no conversion is defined for the current raw value, this and the following fields contain question marks.

If the conversion is not discrete, the message ends after the converted value.

Otherwise, the sixth field is the foreground color and the seventh field is the background color.

3 The dp_server program

```
Usage: dp_server [-daemon [-pid]] [-verbose] [-wkp <port>]
  -daemon     -- run as a daemon.
  -logfile <file> -- write messages to <file> instead of stderr.
  -pid        -- print the daemon's pid to stdout.  '-daemon -pid' are
                 provided so dp_server can be started in a shell
                 script as:
                      d_server -daemon -pid > /tmp/server_pid
                 or
                      pid = `data_server -daemon -pid`
  -verbose    -- generate diagnostic messages.
  -wkp <port> -- listen for client connections over <port>.  The default
                 well-known-port is 7777.
```

Table of Contents